. ...contributing to man's understanding of the environment world

A CONTOUR ROUTINE WITH AUTO-INTERPOLATION

M. WIRTH
SEISMIC DATA LABORATORY

AUGUST 24, 1971

Prepared for
AIR FORCE TECHNICAL APPLICATIONS CENTER
Washington, D.C.

Under
Project VELA UNIFORM

Sponsored by
ADVANCED RESEARCH PROJECTS AGENCY
Nuclear Monitoring Research Office
ARPA Order No. 1714

TELEDYNE GEOTECH
ALEXANDRIA LABORATORIES

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

K

31

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)
TELEDYNE GEOTECH
ALEXANDRIA, VIRGINIA

2a REPORT SECURITY CLASSIFICATION
Unclassified

2b GROUP

3 REPORT TITLE
A CONTOUR ROUTINE WITH AUTO-INTERPOLATION

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)
Scientific

5 AUTHOR(S) (Last name, first name, initial)
Wirth, Mark

6 REPORT DATE
24 August 1971

7a TOTAL NO OF PAGES
29

7b NO OF REFS
1

8a CONTRACT OR GRANT NO.
F33657-72-C-0009

b PROJECT NO
VELA T/2706

ARPA Order No. 1714

d ARPA Program Code No. 2F-10

9a ORIGINATOR'S REPORT NUMBER(S)
272

9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

11 SUPPLEMENTARY NOTES

12 SPONSORING MILITARY ACTIVITY
Advanced Research Projects Agency
Nuclear Monitoring Research Office
Washington, D. C.

13 ABSTRACT

An efficient contour-plotting routine is discussed which is based on a scanning algorithm of Cottafava and LeMoli and employs bi-linear interpolation. An auto-interpolation scheme is developed which automatically adjusts the number of interpolations in any data square to produce smooth line segments. A program listing and examples are given.

14 KEY WORDS
Contour plotting
Plotting

Unclassified
Security Classification

Neither the Advanced Research Projects Agency nor the Air Force Technical Applications Center will be responsible for information contained herein which has been supplied by other organizations or contractors, and this document is subject to later revision as may be necessary. The views and conclusions presented are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency, the Air Force Technical Applications Center, or the U S Government.

A CONTOUR ROUTINE WITH AUTO-INTERPOLATION

SEISMIC DATA LABORATORY REPORT NO. 272

AFTAC Project No.:	VELA T/2706
Project Title:	Seismic Data Laboratory
ARPA Order No.:	1714
ARPA Program Code No.:	2F-10
Name of Contractor:	TELEDYNE GEOTECH
Contract No.:	F33657-72-C-0009
Date of Contract:	01 July 1971
Amount of Contract:	$ 1,290,000
Contract Expiration Date:	30 June 1972
Project Manager:	Royal A. Hartenberger (703) 836-7647

P. O. Box 334, Alexandria, Virginia

ABSTRACT

An efficient contour-plotting routine is discussed which is based on a scanning algorithm of Cottafava and LeMoli and employs bi-linear interpolation. An auto-interpolation scheme is developed which automatically adjusts the number of interpolations in any data square to produce smooth line segments. A program listing and examples are given.

TABLE OF CONTENTS

	Page No.
ABSTRACT	
INTRODUCTION	1
SCANNING ALGORITHM	3
INTERPOLATION	5
AUTO-INTERPOLATION	8
SINGULARITIES AND SADDLE-POINTS	14
RECTANGULAR MESH	15
EXAMPLES	16
REFERENCE	17

LIST OF FIGURES

Figure Title	Figure No.
Intersection flags	1
Interpolation conventions	2
Case I	3
Case II	4
Deviation from a chord	5
Contours in a saddle square	6
Example 1, real data	7
Example 2, random numbers	8

INTRODUCTION

There are many approaches to producing contour maps on a digital computer. Several of these are described by Cottafava and LeMoli (1969). Ideally, a contour routine should be efficient on both computer and plotter, but most seem to possess only one kind of efficiency. For quite a number of reasons, including plotting efficiency, algorithms of the "line-following" type are preferable for use with mechanical plotters. Cottafava and LeMoli present a scanning algorithm of this type which is also very efficient on the computer.

In their program (Cottafava and LeMoli, private communication) they assumed a linear variation between points, but did not do any interpolation in the interior of the data square (defined by four contiguous data points as vertices). Thus a plot produced with their routine consists entirely of straight line segments joined together, the coarseness depending on the spacing of data points. To remedy this, the author developed an autointerpolation scheme employing bi-linear interpolation for use in the interior of the data square. With this scheme, described in the present paper, the number of interpolations used in crossing the data square is automatically adjusted to produce a smooth curve, the number required depending on the curvature of the line segment. This method requires no additional storage and is very fast.

While the interpolation does indeed produce smooth line segments, slope discontinuities sometimes occur on the edges (of the data squares). This is a limitation of the bi-linear interpolation law. An easy and economical solution to this problem is to obtain a finer data mesh by performing a higher-order interpolation before entering the contour routine. This

may also be done when the data are not given as a set of equally-spaced points.

SCANNING ALGORITHM

The scanning algorithm used is that of Cottafava and LeMoli, with several modifications by this author. A more complete discussion than will be given here can be found in the paper cited. For each contour value, the procedure is to scan the entire data array to find which line segments are intersected by the level line and to store the information as flags within the data words themselves. A horizontal and vertical segment are associated with each data point as shown in Figure 1 (the y-axis is given its normal sense here; Cottafava and LeMoli reverse it).

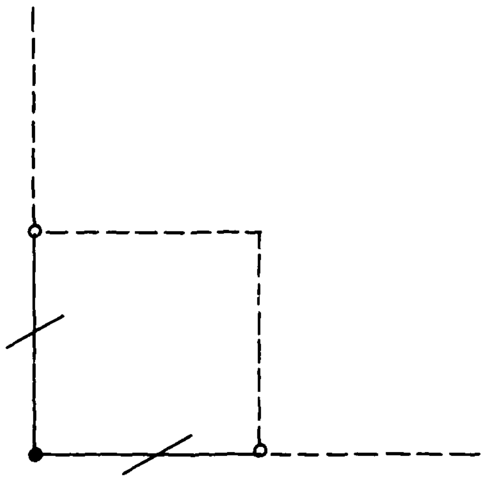

Figure 1. Intersection flags

The flags are stored as bits in the upper part of the integer word, the data being normalized to positive integers by a linear

transformation. (In deference to readers with machines which cannot perform masks, the use of masks in the programs has been limited to an inessential role, where they can easily be replaced by a test and subtract. Where masking statements are available, the flags can be conveniently stored as the least-significant bits of the floating-point mantissa of each data word. This results in a considerable simplification of the program.)

The contours are traced in a second scanning operation, each flag being erased as it is found. This procedure makes it easy to find all the branches of the contour level and is in large part responsible for the efficiency of the program. Each line is traced square by square by a local scan which checks all the edges of the data square in fixed order (counter-clockwise beginning with the right edge) to find the continuation of the line. This procedure runs into trouble only in the case of an interior saddle point (square crossed twice by the same contour), and in this case there is an easy solution based on the interpolation method.

INTERPOLATION

In order to define the behavior of the contour line inside each data square, we must make an assumption about the behavior of the function inside the square. Lacking any special information in the general case, we assume bi-linear variation as the simplest general variation. (Using a higher order interpolation here would also cause serious difficulties with the scanning procedure.) That is, we assume

$$F(x,y) = A + \beta x + \alpha y + \delta xy \tag{1}$$

referred to a local coordinate system with origin at A, as shown in Figure 2.

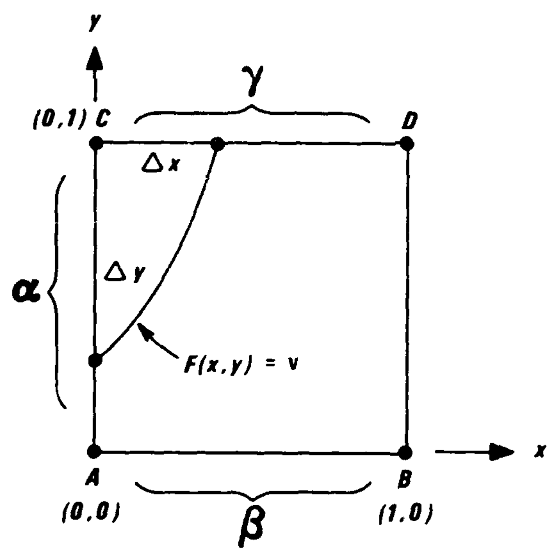

Figure 2. Interpolation conventions

Equation (1) is the Lagrange 2 x 2 interpolation formula and is also equivalent to a Taylor's series expansion, with the assumption of constant first derivatives on each edge. Making our square of unit dimensions, the derivatives are

$$\begin{aligned} \alpha &\equiv C-A \\ \beta &\equiv B-A \\ \gamma &\equiv D-C \\ \delta &\equiv \gamma-\beta \end{aligned} \tag{2}$$

With these definitions, it is easy to see that (1) reduces to the correct values at the corners and reduces to ordinary linear interpolation on each edge. Our contour segment is therefore the locus $F(x,y) = v$, the value of the contour. From (1) we obtain either

$$y = \frac{v-A-\beta x}{\alpha+\delta x} \tag{3}$$

or

$$x = \frac{v-A-\alpha y}{\beta+\delta y} \tag{4}$$

with $0\leq x\leq 1$ and $0\leq y\leq 1$. These expressions are easily computed. In practice, we choose the number of interpolations, subdivide

Δx or Δy, and use either (3) or (4), respectively. It is convenient to use (3) for a line terminating on a vertical edge, and (4) for a line terminating on a horizontal edge.

AUTO-INTERPOLATION

We approximate our ideally smooth curve by a series of chords. If we choose the number of chords in each square so that the maximum deviation from the ideal curve is on the order of the basic plotter increment, then we obtain as smooth a curve as we can with no wasted time. We take as our auto-interpolation criterion, the maximum perpendicular distance from the curve to the straight line between the end points. The perpendicular distance from a point to a line is

$$\cdot(x) = (y - mx - b)/\sqrt{m^2+1} \qquad (5)$$

from elementary geometry. We consider the case of a segment terminating on the <u>left</u> edge of the square, and for this case there are just two distinct possibilities, as shown in Figures 3 and 4. All other possibilities can be found by reflections and a rotation.

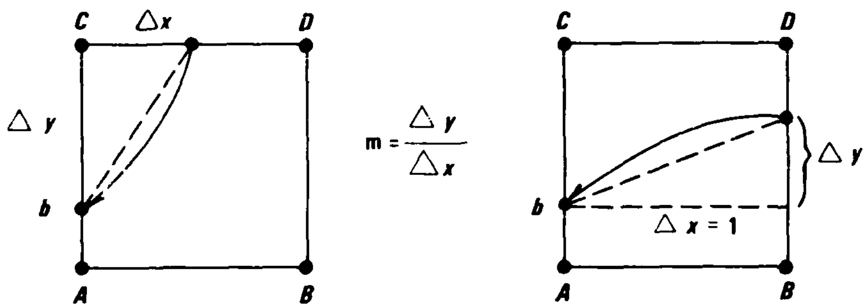

Figure 3. Case I

Figure 4. Case II

For case I the slope is

$$m_1 = \frac{v-C}{C-A} \Big/ \frac{v-C}{C-D} = -\gamma/\alpha$$

For case II,

$$m_2 = \frac{v-A-\beta}{\alpha+\delta} - \frac{v-A}{\alpha} = - \frac{\delta(v-A) + \alpha\beta}{\alpha(\alpha+\delta)}$$

For both cases the intercept is $b = (v-A)/\alpha$. The "brute force" calculation of the maximum of (5) is messy, but a transformation simplifies the algebra. Define

$$\eta_1 \equiv \frac{\delta(v-C)}{\alpha\gamma}$$

$$\eta_2 \equiv \delta/\alpha \tag{6}$$

$$\eta \equiv \frac{\delta}{\alpha}\,\Delta x \text{ where } \Delta x = \begin{cases} (v-C)/\gamma, & \text{case I} \\ 1, & \text{case II} \end{cases}$$

Then

$$\frac{\delta(v-A)}{\alpha} + \beta = \delta\left[1 + \frac{v-C}{\alpha}\right] + \beta = \gamma[1 + \eta_1]$$

which leads to the relation

$$m_2 \, (1+\eta_2) = m_1(1+\eta_1) \tag{7}$$

It can also be shown that

$$y-b = \frac{m_1(1+\eta_1)}{1+\eta_2 x} \, x$$

and

$$y' = \frac{dy}{dx} = \frac{m_1(1+\eta_1)}{(1+\eta_2 x)^2}$$

The location of the maximum of (5) is given by the condition

$$\varepsilon'(\hat{x}) = 0 = y'(\hat{x}) - m$$

(hats will be used to refer to the maximum), i.e.,

$$\frac{m_1(1+\eta_1)}{(1+\eta_2\hat{x})^2} = m \tag{8}$$

Thus

$$\hat{y} - b = m(1+\eta_2\hat{x})\hat{x}$$

and

$$\sqrt{m^2+1}\ \hat{\varepsilon} = \hat{y} - b - m\hat{x} = m\eta_2\hat{x}^2$$

or

$$\hat{\varepsilon} = \frac{m}{\sqrt{m^2+1}}\ \eta_2\ \hat{x}^2 \tag{9}$$

where $\hat{x}$ is found from (8). For case I, $m = m_1$, and

$$\hat{x}_1 = \frac{1}{\eta_2}\ [\sqrt{1+\eta_1} - 1]$$

For case II, $m = m_2$, and using (7) in (8) gives

$$\hat{x}_2 = \frac{1}{\eta_2}\ [\sqrt{1+\eta_2} - 1]$$

which has the same form. Putting these results in (9) gives

$$\hat{\varepsilon} = \frac{m}{\sqrt{m^2+1}} \frac{1}{\eta^2} [\sqrt{1+\eta} - 1]^2$$

for both cases. This is conveniently written in the form

$$\hat{\varepsilon} = \frac{\Delta x \ \Delta y}{\sqrt{\Delta x^2+\Delta y^2}} \{\frac{1}{\eta} (\sqrt{1+\eta} - 1)^2\} \tag{10}$$

Since (10) is symmetric in x and y, it can be easily seen to apply to the case of a segment terminating on the bottom edge of the square also, provided that we define $\eta \equiv \delta\Delta y/\beta$ in this case. Thus all possible cases are contained in (10). A bound can be placed on $\hat{\varepsilon}$ by considering $\eta\to\infty$ and $\Delta x\to\Delta y\to 1$, namely $\hat{\varepsilon}<1/\sqrt{2}$, which agrees with geometrical intuition.

Knowing how to calculate the maximum deviation of the curve from the straight line between endpoints, we use this to estimate the number of segments needed to approximate the curve to any desired accuracy. To do this, we consider the case of a circular arc, Figure 5.

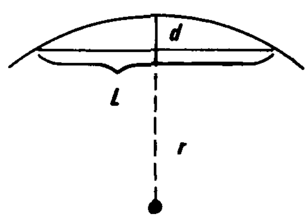

Figure 5. Deviation from a chord

It is easily shown that $d \simeq L^2/8r$, provided $L/r \ll 1$, the important point being the proportionality $d \propto L^2$. Since $L \propto 1/N$, approximately, where N is the number of segments needed in that square, we take

$$N = 1 + \sqrt{|\varepsilon|/d}$$

where d equals the allowable deviation. For plotters with a 2.5 mil increment, d = .001" is satisfactory. This auto-interpolation scheme appears to work quite well.

SINGULARITIES AND SADDLE-POINTS

Evidently something peculiar happens with (10) if $\eta+1<0$. This can be seen to be connected with a singularity in y or x, equation (3) or (4). The existence of a singularity in $y(x)$ within the interval $0\leq x\leq 1$ is implied by the condition that (D-B) and α have opposite signs, and the existence of a singularity in $x(y)$ within the interval $0\leq y\leq 1$ is implied by the condition that γ and β have opposite signs. If both singularities exist, then the square has an interior saddle point located at the intersection. An example is shown in Figure 6. A very convenient criterion for making connections in a saddle square is that contours should never cross a singularity. The condition $\eta+1<0$ can be easily shown to imply a wrong connnection in a saddle square. By far the easiest solution to this problem is just to check for a wrong connection and to resume scanning the square if it exists. At most three tries will be necessary to make the correct connection, and since saddle-points should be relatively rare, this is a small price to pay for such a simple procedure that guarantees correct connections.

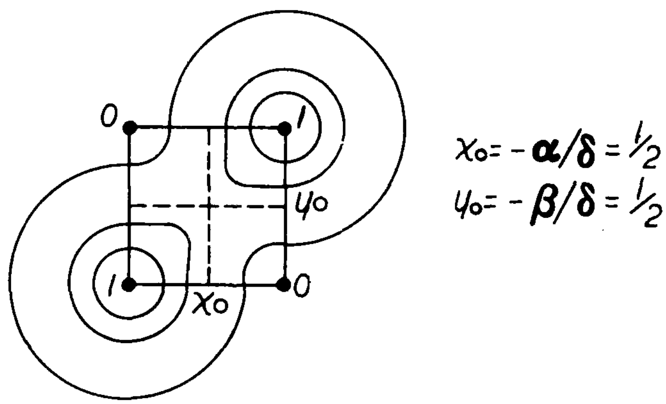

Figure 6. Contours in a saddle square

RECTANGULAR MESH

Should it be desired to plot data cells as rectangles instead of squares, this is easily done by stretching one axis in the calls to PLOT. Defining r as the ratio of x to y scale factors, i.e. the rectangle has length r in the x-direction and 1 in the y-direction, elementary trigonometry gives for the modified deviation

$$\tilde{\varepsilon} = \varepsilon\sqrt{\frac{1+r^2m^2}{1+m^2}} = \varepsilon\sqrt{\frac{\Delta x^2+r^2\Delta y^2}{\Delta x^2+\Delta y^2}}$$

EXAMPLES

Two simple examples of plots produced by the routine are shown in Figures 7 and 8. Figure 7 was produced from real, deterministic, data, and Figure 8 from random numbers. Data points are marked by ticks along the borders of the plots. Each plot is based on only 24 data points, and the large number of slope discontinuities indicates the need for a more refined data mesh.

REFERENCE

Cottafava, G. and LeMoli, G., 1969, Automatic contour map: Comm. ACM 12(July), p. 386-391.

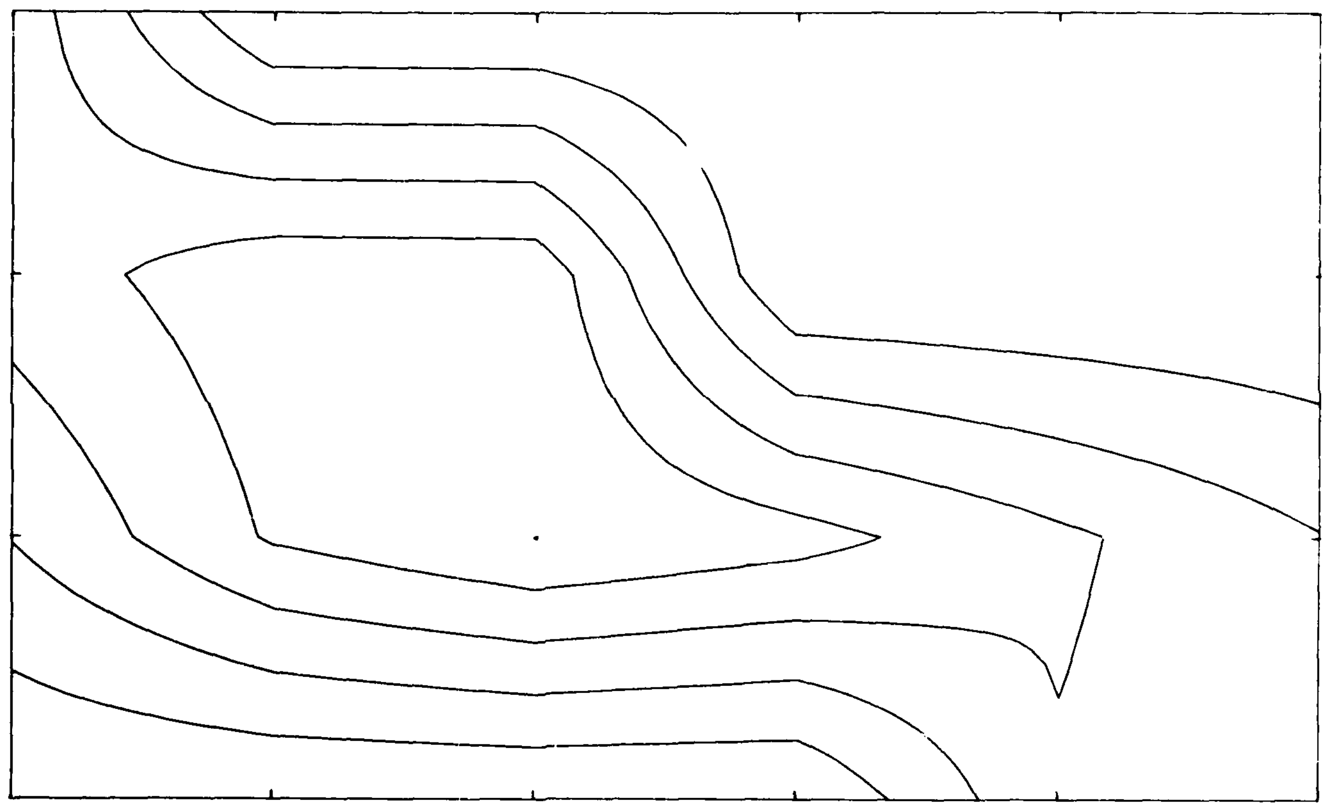

Figure 7. Example 1, real data

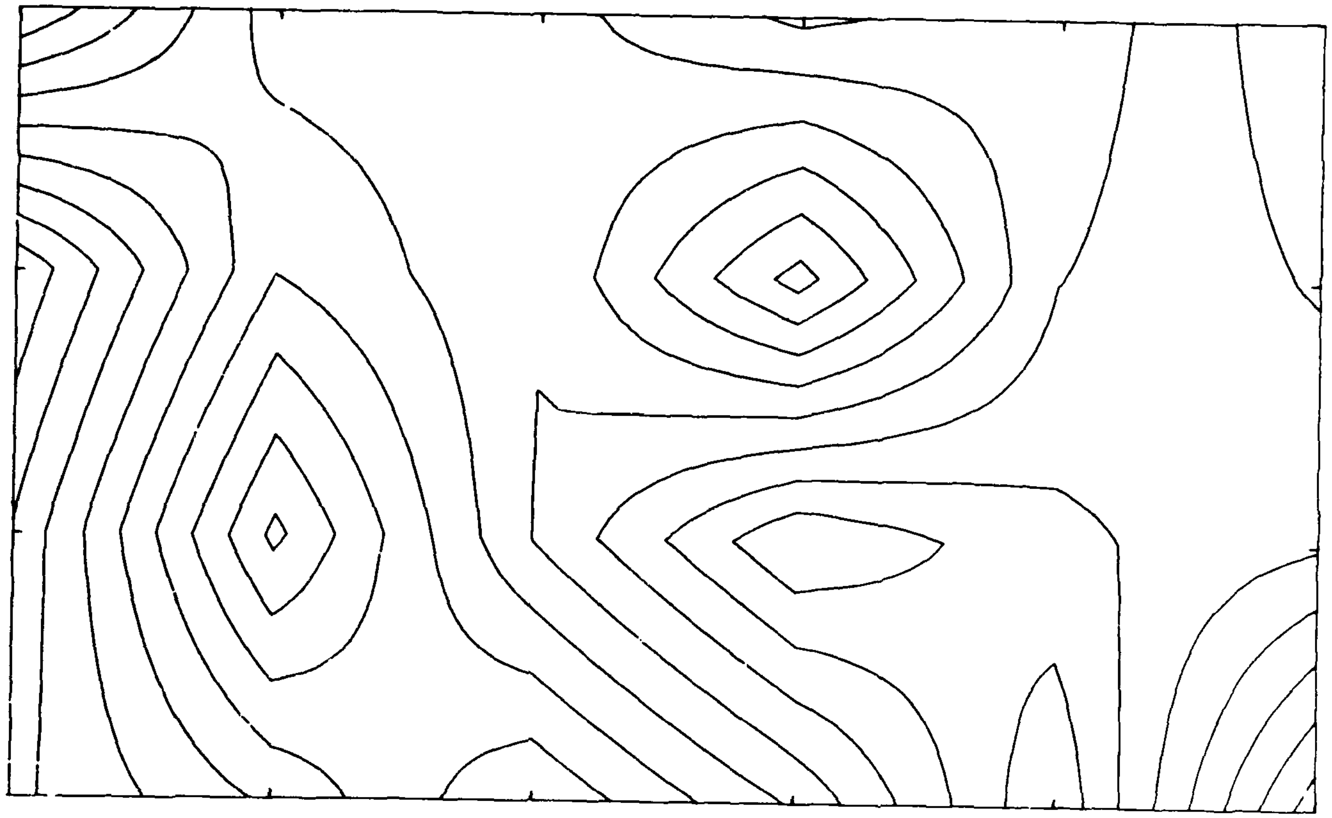

Figure 8. Example 2, random numbers

APPENDIX

PROGRAM LISTING

Written in FORTRAN-63, a programming language of the CDC 1604 computer. Integer words assumed at least 33 bits long.

```
      SUBROUTINE CONTOR( M,N,F, K,C, ND,ND2,IV,IA, HEIGHT,DEV )
      DIMENSION  C(K), F(ND,M), IV(ND2,M), IA(M)
      DATA (IHOR = 10000000000B), (IVER = 20000000000B)
C
C     PLOT CONTOURS  F(I,J) = F(Y,X) = C,  I=1,N  $  J=1,M
C     IV $ IA ARE AUXILIARY ARRAYS.  MAY EQUIVALENCE (F,IV).
C     MUST HAVE  ND2 ≥ N+2
C     *DEV* IS AUTO-INTERPOLATION PARAMETER = APPROX. ALLOWABLE DEV.
C     (INCHES).  SUGGEST  DEV = .001
C     ALGORITHM BY COTTAFAVA $ LEMOLI, POLITECNICO DI MILANO.
C     TRANSCRIBED $ REVISED BY MARK WIRTH, JANUARY 1971.
C     INTERPOLATION ALGORITHM BY MARK WIRTH
C
C     NORMALIZE DATA
      FMAX  = F          $    FMIN = 0.
      DO 5 J = 1,M
      DO 5 I = 1,N
      IF( F(I,J).GT.FMAX ) 1,2
    1 FMAX = F(I,J)      $    GO TO  5
    2 IF( F(I,J).LT.FMIN ) 3,5
    3 FMIN = F(I,J)
    5 CONTINUE
      A = 1E+8/(FMAX-FMIN)                   $     B = -FMIN*A +.5
      JR = M1 = M + 1    $    N1 = N + 1     $     NP2 = N + 2
      DO 8 J = 1,M
      DO 7 I = 1,N
    7 IV(I+1,JR) = A*F(I,JR-1) + B
      IV(1,JR) = IV(NP2,JR) = 0
    8 JR = JR - 1
      CALL ERASE( NP2,IV )
      CALL ERASE( NP2,IV(1,M+2) )
      FAC = HEIGHT/(N-1)                     $     DEVS = DEV / FAC
      CALL FACTOR( FAC )
C
C     LOOP OVER CONTOUR LEVELS
      DO 500 L = 1,K
      VO = IVOL = A*C(L) + B
C     PRELIMINARY SCAN.  SET FLAGS
      DO 20 J = 2,M1
      IA(J) = IV(2,J) - IVOL
      IF( IA(J) ) 20,10
   10 IA(J) = 1          $    IV(2,J) = IV(2,J) + 1
   20 CONTINUE
      DO 100 I = 2,N1
      II = I + 1
      DO 100 J = 2,M1
      IF( I.EQ.N1 ) 60,60
   30 IC = IV(II,J) - IVOL
      IF( IC ) 40,35
   35 IC = 1             $    IV(II,J) = IV(II,J) + 1
   40 IF( XSIGNF(1,IA(J))*IC ) 50,50,60
   50 IV(I,J) = IV(I,J) + IVER
   60 IF( J.EQ.M1 ) 100,70
   70 IF( XSIGNF(1,IA(J))*IA(J+1) ) 80,80,100
   80 IV(I,J) = IV(I,J) + IHOR
  100 IA(J) = IC
```

```
C
C     PICK UP BRANCHES OF CONTOUR LEVEL
      DO 200 J = 2,M
C     BOTTOM EDGE
      IIV = IV(2,J).AND..NOT.IVER
      IF( IIV.GE.IHOR ) 120,130
  120 CALL BRANCH( 2,J,1,1, ND2,IV,VO,DEVS )
C     TOP EDGE
  130 IF( IV(N1,J).GE.IHOR ) 140,200
  140 CALL BRANCH( N1,J,1,2, ND2,IV,VO,DEVS )
  200 CONTINUE
C     RIGHT EDGE
      DO 300 I = 2,N
      IF( IV(I,M1).GE.IVER ) 250,300
  250 CALL BRANCH( I,M1,2,3, ND2,IV,VO,DEVS )
  300 CONTINUE
C     LEFT EDGE & INTERIOR
      DO 500 J = 2,M
      KK = 2 / J
      DO 500 I = 2,N
      IF( IV(I,J).LT.IVER ) 500,350
  350 CALL BRANCH( I,J,2,KK, ND2,IV,VO,DEVS )
  500 CONTINUE
      CALL FACTUR( 1. )
      RETURN
      END
```

```
      SUBROUTINE BRANCH( IREM,JREM, LISP,K, ND,IV,VO,DEVS )
      DIMENSION  IV(ND,1)
      DATA (IHOR= 1000000000B),(IVER= 20000000000B),(IVH= 30000000000B)
C     FOLLOW ONE BRANCH OF CONTOUR LEVEL
C
      IPEN = 3          $      IP = I = IREM       $     JP = J = JREM
      XP = J - 2        $      YP = I - 2          $     IT = 0
      MISP = LISP       $      ASSIGN 1 TO JAIL    $     GO TO  200
    1 IPEN = 2
      GO TO (5,4,3), K
    3 JS = J = J - 1                     $      GO TO  5
    4 IS = I = I - 1
    5 IF( IV(I,J+1).GE.IVER ) 10,15
   10 J = J+1     $     MISP = 2    $     ASSIGN 5 TO JAIL     $     GO TO  200
   15 IBV = IV(I+1,J).AND..NOT.IVER
      IF( IBV.GE.IHOR ) 20,25
   20 I = I+1     $     MISP = 1    $     ASSIGN 5 TO JAIL     $     GO TO  200
   25 IF( IV(I,J).GE.IVER ) 30,35
   30 MISP = 2          $      ASSIGN 3 TO JAIL    $     GO TO  200
   35 IF( IV(I,J).GE.IHOR ) 40,90
   40 MISP = 1          $      ASSIGN 4 TO JAIL    $     GO TO  200
   90 IF( K ) 100,190
  100 RETURN
C
C     REMOVE FLAGS, INTERPOLATE, AND PLOT POINT
  190 I = IREM          $      J = JREM                 $     IV(I,J) = IV(I,J)+IVER
      MISP = 2          $      ASSIGN 100 TO JAIL
  200 XO = J - 2        $      YO = I - 2
      JQ = JP           $      IQ = IP
      IF( J.EQ.JP ) 210,220
  210 JQ = JP + 1
  220 IF( I.EQ.IP ) 230,240
  230 IQ = IP + 1
  240 A = IA = IV(I,J).AND..NOT.IVH
      B = IB = IV(I,JQ).AND..NOT.IVH
      C = IC = IV(IQ,J).AND..NOT.IVH
      D = ID = IV(IQ,JQ).AND..NOT.IVH
      ALF = C - A       $      BET = B - A          $     DEL = D-C - BET
      GO TO (250,260), MISP
  250 XF = (VO-A)/BET + XO        $     DX = XF - XP         $     YF = YO
      SIGN = IQ - I               $     DY = Y = SIGN*(YP-YO)
      ETA = DY*DEL / BET          $     GO TO  270
  260 YF = (VO-A)/ALF + YO        $     DY = YF - YP         $     XF = XO
      SIGN = JQ - J               $     DX = X = SIGN*(XP-XO)
      ETA = DX*DEL / ALF
  270 IF( ETA ) 280,400
  280 ETAP = ETA + 1.
C     SADDLE-POINT TEST
      IF( ETAP ) 290,295,300
  290 I = IS          $     J = JS                   $     IT = IT + 1
      GO TO (15,25,35), IT
  295 EPS = 1.        $     GO TO  310
  300 EPS = DX*DY*(SQRTF(ETAP)-1.)**2 / (ETA*SQRTF(DX*DX+DY*DY))
  310 NI = 1.5 + SQRTF(ABSF(EPS)/DEVS)
      ANI = NI *.9999               $     IT = 0
      GO TO (320,350), MISP
```

```
320 DY = Y / FNI
    F = VC-A - ALF*Y              $     DF = -ALF*DY
    G = BET + DEL*Y               $     DG = DEL*DY
330 Y = Y - DY
    IF( Y ) 400,400,340
340 F = F - DF     $     G = G - DG          $     X = F/G
    CALL PLOT( X0+X,Y0+SIGN*Y, IPEN )
    GO TO  330
350 DX = X / FNI
    F = VC-A - BET*X              $     DF = -BET*DX
    G = ALF + DEL*X               $     DG = DEL*DX
360 X = X - DX
    IF( X ) 400,400,370
370 F = F - DF     $     G = G - DG          $     Y = F/G
    CALL PLOT( X0+SIGN*X,Y0+Y, IPEN )
    GO TO  360
400 IV(I,J) = IV(I,J) - IHOR*MISP
    IS = IP = I    $     JS = JP = J   $     XP = XF      $     YP = YF
    CALL PLOT( XP,YP, IPEN )
    GO TO JAIL,  (1,3,4,5,100)
    END
```